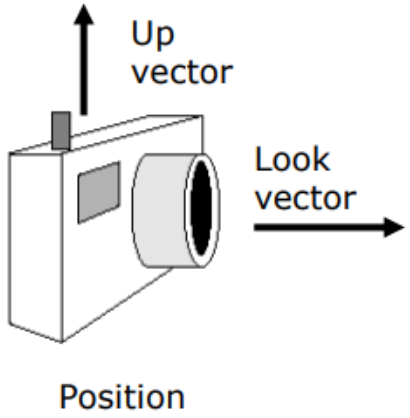
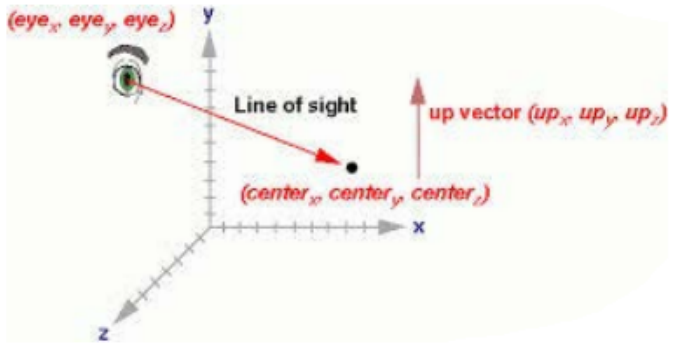
EPL 426

Lab 5 – OpenGL – Camera, Lights and more

Andreas Andreou

Camera in OpenGL





- Χρειαζόμαστε 3 πράγματα:
  - Θέση της camera (position)
  - Look Vector (που κοιτάζει)
  - Camera’s Orientation (Up Vector)

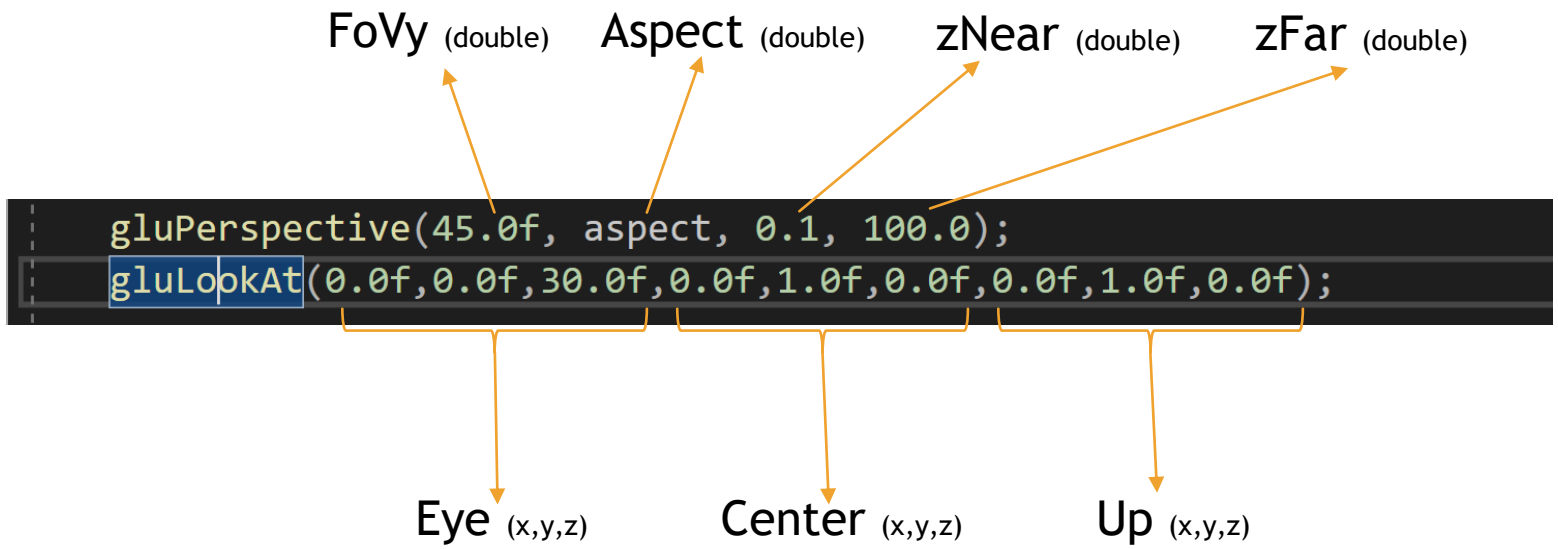
Camera

```
void gluLookAt(  GLdouble eyeX,
                 GLdouble eyeY,
                 GLdouble eyeZ,
                 GLdouble centerX,
                 GLdouble centerY,
                 GLdouble centerZ,
                 GLdouble upX,
                 GLdouble upY,
                 GLdouble upZ  );
```

Camera

- **eyeX**, **eyeY**, **eyeZ**: Specifies the position of the eye point.
- **centerX**, **centerY**, **centerZ**: Specifies the position of the reference point.
- **upX**, **upY**, **upZ**: Specifies the direction of the up vector.
- gluLookAt(0.0f,0.0f,30.0f,0.0f,1.0f,0.0f,0.0f,1.0f,0.0f);

Camera


FoVy (double)
Aspect (double)
zNear (double)
zFar (double)
gluPerspective(45.0f, aspect, 0.1, 100.0);
gluLookAt(0.0f,0.0f,30.0f,0.0f,1.0f,0.0f,0.0f,1.0f,0.0f);
Eye (x,y,z)
Center (x,y,z)
Up (x,y,z)

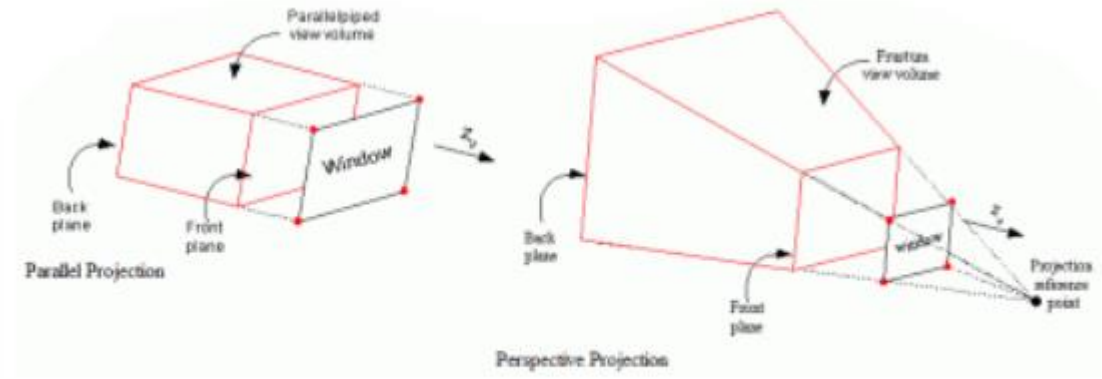
Camera

- Τώρα δοκιμάστε να αλλάξετε τις μεταβλητές της gluLookat για να πάρετε την εικόνα της επόμενης διαφάνειας. Προτού δοκιμάσετε διάφορα πράγματα, δέστε στη θεωρία τη θέση που πρέπει να βρίσκεται η camera σας, το που πρέπει να βλέπει και που πρέπει να είναι το Up Vector της.


```
gluLookAt(10.0f,- 10.0f, 60.0f, 10.0f, -10.0f, 0.0f, 1.0f, 1.0f, 0.0f);
```

Projections

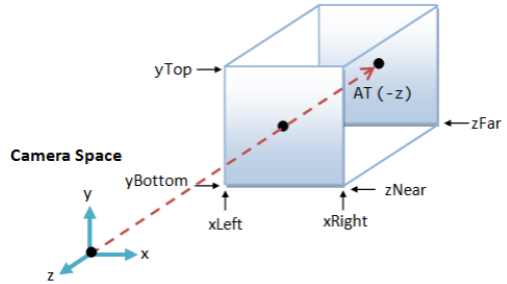
- Έχουμε δύο είδη προβολής:
  - Perspective
  - Orthografic (parallel)



Orthographic & Perspective Projection

```
void glOrtho(GLdouble left,
             GLdouble right,
             GLdouble bottom,
             GLdouble top,
             GLdouble nearVal,
             GLdouble farVal);
```

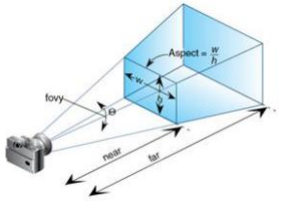
- **left**, **right**: Specify the coordinates for the left and right vertical clipping planes.
- **bottom**, **top**: Specify the coordinates for the bottom and top horizontal clipping planes.
- **nearVal**, **farVal**: Specify the distances to the nearer and farther depth clipping planes. These values are negative if the plane is to be behind the viewer.

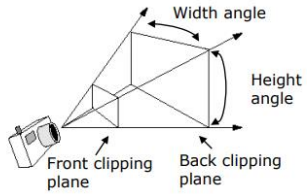


Orthographic Projection: Camera positioned infinitely far away at $z = \infty$

```
void gluPerspective(GLdouble fovy,
                    GLdouble aspect,
                    GLdouble zNear,
                    GLdouble zFar);
```

- **fovy**: Specifies the field of view angle, in degrees, in the y direction.
- **aspect**: Specifies the aspect ratio that determines the field of view in the x direction. The aspect ratio is the ratio of x (width) to y (height).
- **zNear**: Specifies the distance from the viewer to the near clipping plane (always positive).
- **zFar**: Specifies the distance from the viewer to the far clipping plane (always positive).





Lights

- Στην OpenGL πέρα από την ενεργοποίηση του φωτισμού, χρειάζεται να θέσουμε τις φωτεινές μας πηγές, τον τύπο τους και την τοποθεσία που βρίσκονται ώστε να φωτίζουν ανάλογα τα αντικείμενα που έχουμε στη σκηνή μας
- Συνεπώς, πέρα από την αρχικοποίηση και ενεργοποίηση του φωτισμού, χρειάζεται να δημιουργήσουμε τις φωτεινές πηγές μας και να τις τοποθετήσουμε κατάλληλα στη σκηνή μας
- Αυτό θα γίνει με την συνάρτηση glLightfv

Lights

```
void glLightfv(GLenum       light,
               GLenum       pname,
               const GLfloat * params);
```

- ***light:*** Specifies a light. The number of lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL_LIGHT i, where i ranges from 0 to the value of GL_MAX_LIGHTS - 1.
- ***pname*** Specifies a light source parameter or *light*.
  GL_AMBIENT,
  GL_DIFFUSE,
  GL_SPECULAR,
  GL_POSITION,
  GL_SPOT_CUTOFF,
  GL_SPOT_DIRECTION,
  GL_SPOT_EXPONENT,
  GL_CONSTANT_ATTENUATION,
  GL_LINEAR_ATTENUATION,
  GL_QUADRATIC_ATTENUATION
- ***params:*** Specifies a pointer to the value or values that parameter *pname* of light source *light* will be set to.

https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glLight.xml

Material

```
void glMaterialfv(GLenum        face,
                  GLenum        pname,
                  const GLfloat * params);
```

- ***face:*** Specifies which face or faces are being updated. Must be one of GL_FRONT, GL_BACK, or GL_FRONT_AND_BACK.
- ***pname****:* Specifies the material parameter of the face or faces that is being updated. Must be one of

  GL_AMBIENT

  GL_DIFFUSE

  GL_SPECULAR

  GL_EMISSION

  GL_SHININESS

  GL_AMBIENT_AND_DIFFUSE

  GL_COLOR_INDEXES
- ***params****:* Specifies a pointer to the value or values that *pname* will be set to.

https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glMaterial.xml

Lighting & Material

```
glEnable(GL_LIGHT0);
//glEnable(GL_LIGHT1);
glEnable(GL_LIGHTING);
```

Default light

Extra light

Set material specular behavior

```
// Set Specular
GLfloat matSpec[] = { 0.1, 0.1,0.1,1 };
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, matSpec);
GLfloat shininess[] = { 64 };
glMaterialfv(GL_FRONT, GL_SHININESS, shininess);
```

Set material specular shininess

```
GLfloat lightOn[4] = { 1,1,1,1 };
GLfloat lightAmbientOn[4] = { 0.1,0.1,0.1,1 };
GLfloat lightOff[4] = { 0,0,0,0 };
switch (temp3)
{
    case 1:
    {
        printf("AMBIENT only\n");
        glLightfv(GL_LIGHT1, GL_AMBIENT, lightAmbientOn);
        glLightfv(GL_LIGHT1, GL_DIFFUSE, lightOff);
        glLightfv(GL_LIGHT1, GL_SPECULAR, lightOff);
        break;
    }
    case 2:
    {
        printf("DIFFUSE only\n");
        glLightfv(GL_LIGHT1, GL_AMBIENT, lightOff);
        glLightfv(GL_LIGHT1, GL_DIFFUSE, lightOn);
        glLightfv(GL_LIGHT1, GL_SPECULAR, lightOff);
        break;
    }
```

Set light ambient color

Set light diffuse color

Set light specular color

Find more info:

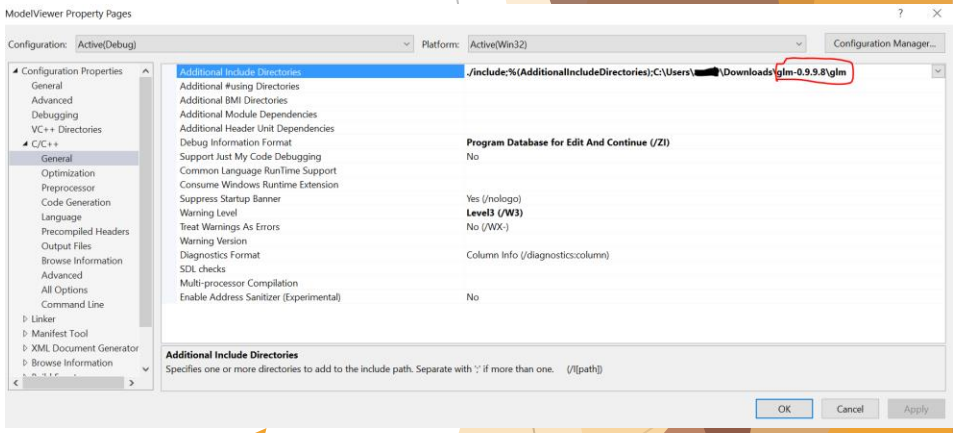
https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glLight.xml
https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glMaterial.xml
https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glColorMaterial.xml

Moving Light

```
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
```

```
// Make and rotating light
if (movingLightFlag)
{
    angle = angle + 0.1;
    glm::mat4 myMatrix =
    {
        cos(angle),0,sin(angle),0,
        0,1,0,0,
        -sin(angle),0,cos(angle),0,
        0,0,0,1
    };

    glm::vec4 myVector = { 0.0f, 0.0f, -4.0f, 1 };
    glm::vec4 transformedVector = myMatrix * myVector;
    GLfloat lightpos[4] = { transformedVector[0] ,transformedVector[1] ,transformedVector[2] ,transformedVector[3] };
    glLightfv(GL_LIGHT1, GL_POSITION, lightpos);
    //printf("%f , %f, %f, %f\n", myMatrix[0][0], myMatrix[0][1], myMatrix[1][0], myMatrix[1][1]);
    //printf("%f , %f, %f, %f\n", transformedVector[0], transformedVector[1], transformedVector[2], transformedVector[3]);
    glColor3f(1.0f, 0.0f, 0.0f); // red
    glTranslatef(transformedVector[0], transformedVector[1], transformedVector[2]);
    glutSolidCube(1);
}
```



OpenGl Transformations: https://learnopengl.com/Getting-started/Transformations

GLM Library: https://glm.g-truc.net/0.9.9/index.html

GLM Library Download: https://github.com/g-truc/glm/releases/tag/0.9.9.8

GLM Library Install: https://stackoverflow.com/questions/17905794/how-to-setup-the-glm-library-in-visual-studio-2012

GLM Library Manuall: glm-0.9.9.8\glm\doc\manual.pdf

Window OpenGL

- `glutInitWindowPosition(100, 100); // Window Position`
- `glutInitWindowSize(500, 500); // Window Size If We Start In Windowed Mode`
- `glutCreateWindow("EPL426"); // Window Title`
- `glutGet(GLUT_SCREEN_WIDTH) & glutGet(GLUT_SCREEN_HEIGHT)`
  `// Get screen width and height`

```
int screenSize[] = { glutGet(GLUT_SCREEN_WIDTH), glutGet(GLUT_SCREEN_HEIGHT) };
int windowsSize[] = { 1000,1000 };
glutInitWindowPosition((screenSize[0]/2) - (windowsSize[0]/2), (screenSize[1] / 2) - (windowsSize[1] / 2));
glutInitWindowSize(windowsSize[0], windowsSize[1]);
glutCreateWindow("EPL426 - 1st Assignment");
```

Keyboard in OpenGL

```
// Our Keyboard Handler (Normal Keys)
void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
        case 'w':
            break;
        case 's':
            break;
        case 'd':
            temp += 1;
            break;
        case 'a':
            temp -= 1;
            break;

        case 27:        // When Escape Is Pressed...
            exit(0);    // Exit The Program
            break;      // Ready For Next Case
        default:
        break;
    }
    glutPostRedisplay();
}
```

Mouse in OpenGL

- Δημιουργήστε μια νέα συνάρτηση (mouseButton) πάνω από τη main

```
void mouseButton(int button, int state, int x, int y) {
    // only start motion if the left button is pressed
    if (button == GLUT_LEFT_BUTTON) {
        if (state == GLUT_UP) // state = GLUT_DOWN
        {
            temp2 = 0;
            //Do something
        }
        else // state = GLUT_DOWN
        {
            temp2 = 45;
            // Do something Else
        }
    }
    else if (button == GLUT_RIGHT_BUTTON)
    {
        if (state == GLUT_DOWN) // state = GLUT_DOWN
        {
            temp2 -= 1;
            //Do something
        }

    }
}
```

- Έπειτα μέσα στη Main, βάρτε την ακόλουθη γραμμή κώδικα:
  - glutMouseFunc(mouseButton);

Mouse in OpenGL (Motion)

- Για ανίχνευση της κίνησης του Mouse, δημιουργήστε μια νέα συνάρτηση (mouseMove) πάνω από τη main:

```
void mouseMove(int x, int y) {
    // Do what ever you want here, x & y is the position of the mouse
    printf("Mouse position is: %i in X axis and %i in Y axis\n", x, y);
}
```

- Έπειτα μέσα στη Main, βάρτε την ακόλουθη γραμμή κώδικα:
  - glutMotionFunc(mouseMove);
- Γενικά αυτή η συνάρτηση καλείται όταν είναι πατημένο το αριστερό ή το δεξί κουμπί του mouse